JÜLICH
FORSCHUNGSZENTRUM

HPC I/O at Large Scale with SIONlib and Spindle

Wolfgang Frings

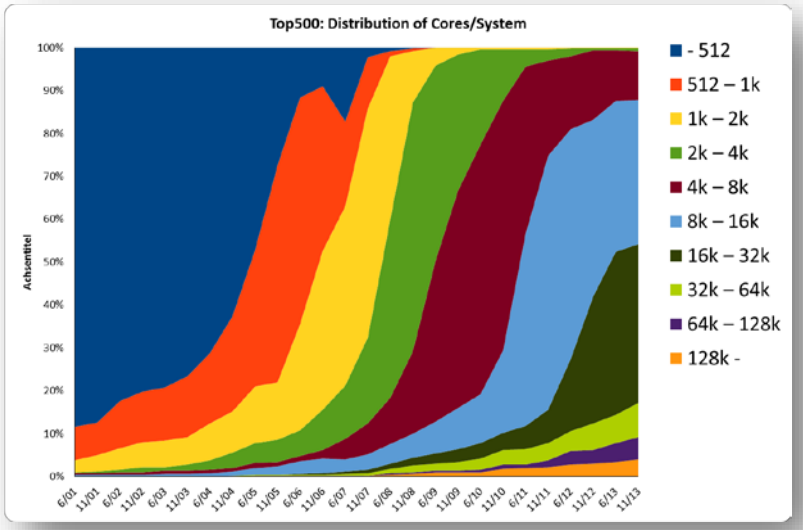
W.Frings@fz-juelich.de

Jülich Supercomputing Centre

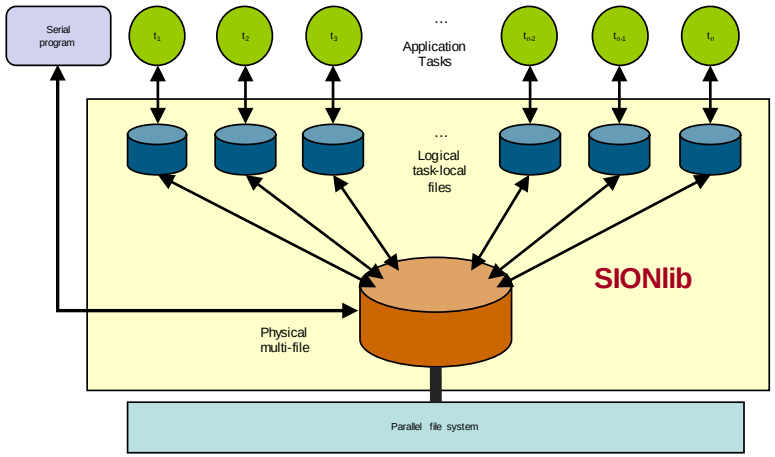
11th Workshop of the INRIA-Illinois-ANL
Joint Laboratory for PetaScale Computing
INRIA, Sophia Antipolis, 9. -11. June 2014

Introduction

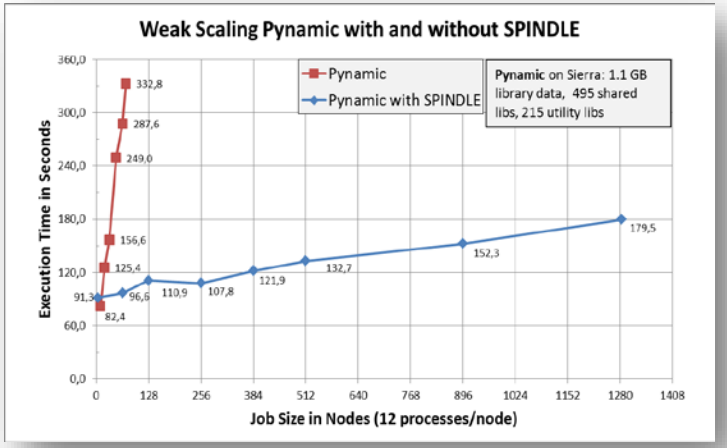
- Increasing number of cores per application
- HPC I/O may become a bottleneck at large scale
- Two tools to support application at that scale: SIONlib & Spindle



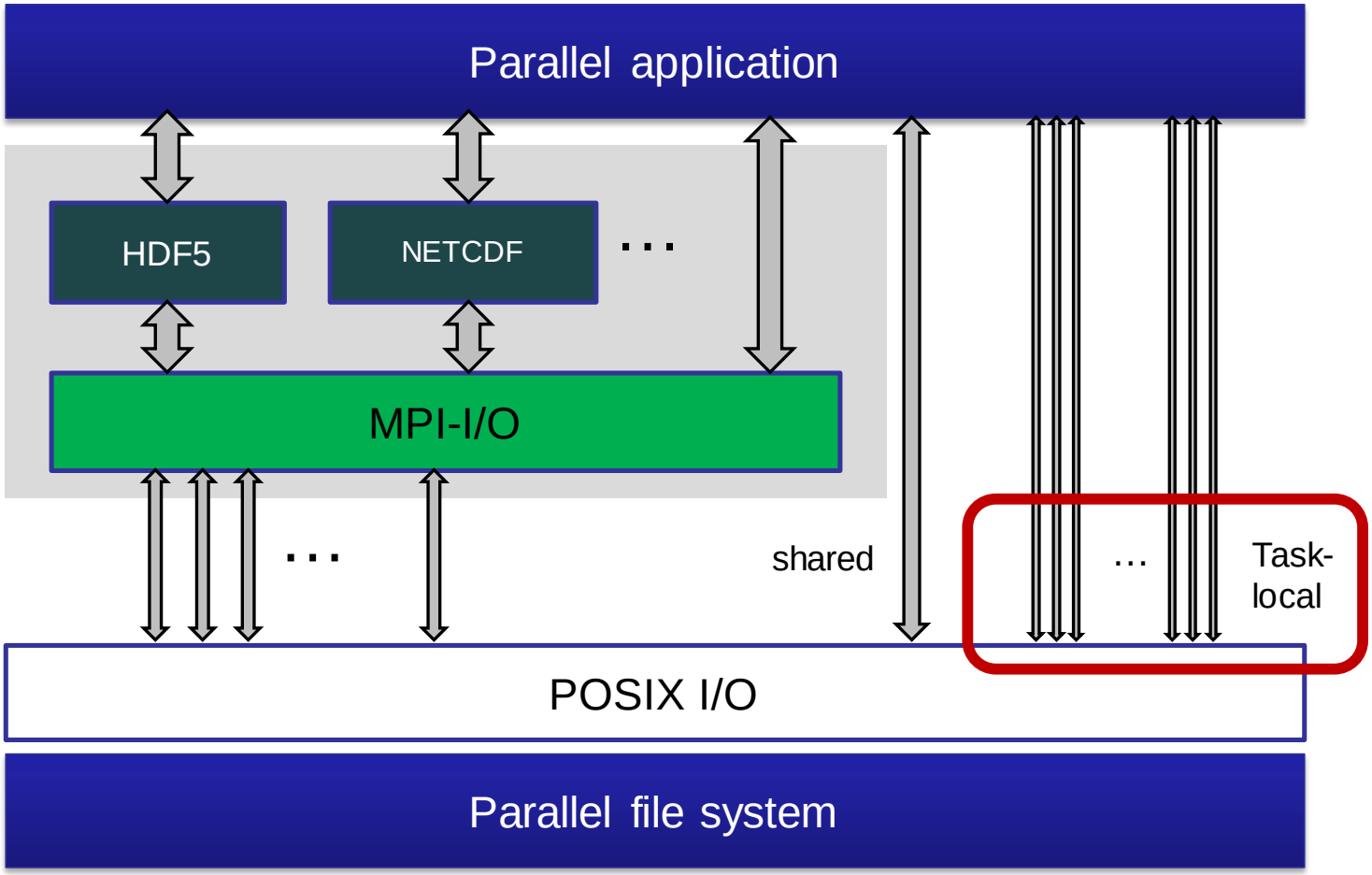
SIONlib: Parallel I/O to task-local files at large scale



SPINDLE: Scalable Parallel Input Network for Dynamic Load Environments



SIONlib: Application View to Parallel I/O



Parallel Task-local I/O at Large Scale

Usage Fields:

- Check-point files, restart files
- Result files, post-processing
- Parallel Performance-Tools

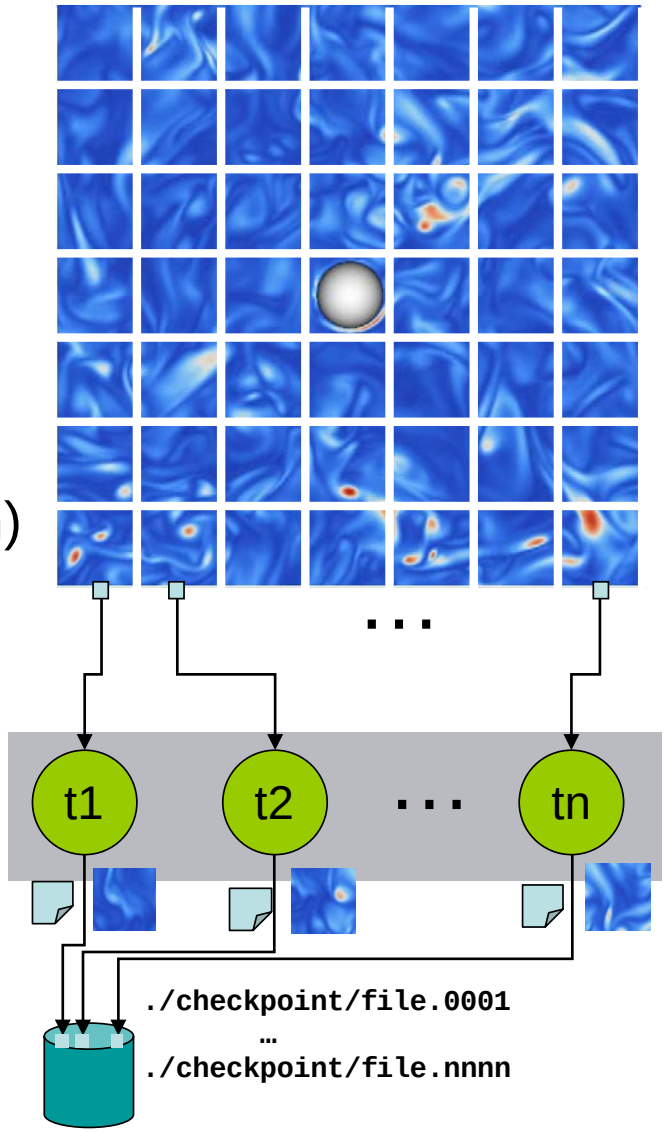
Data types:

- Simulation data (domain-decomposition)
- Trace data (parallel performance tools)

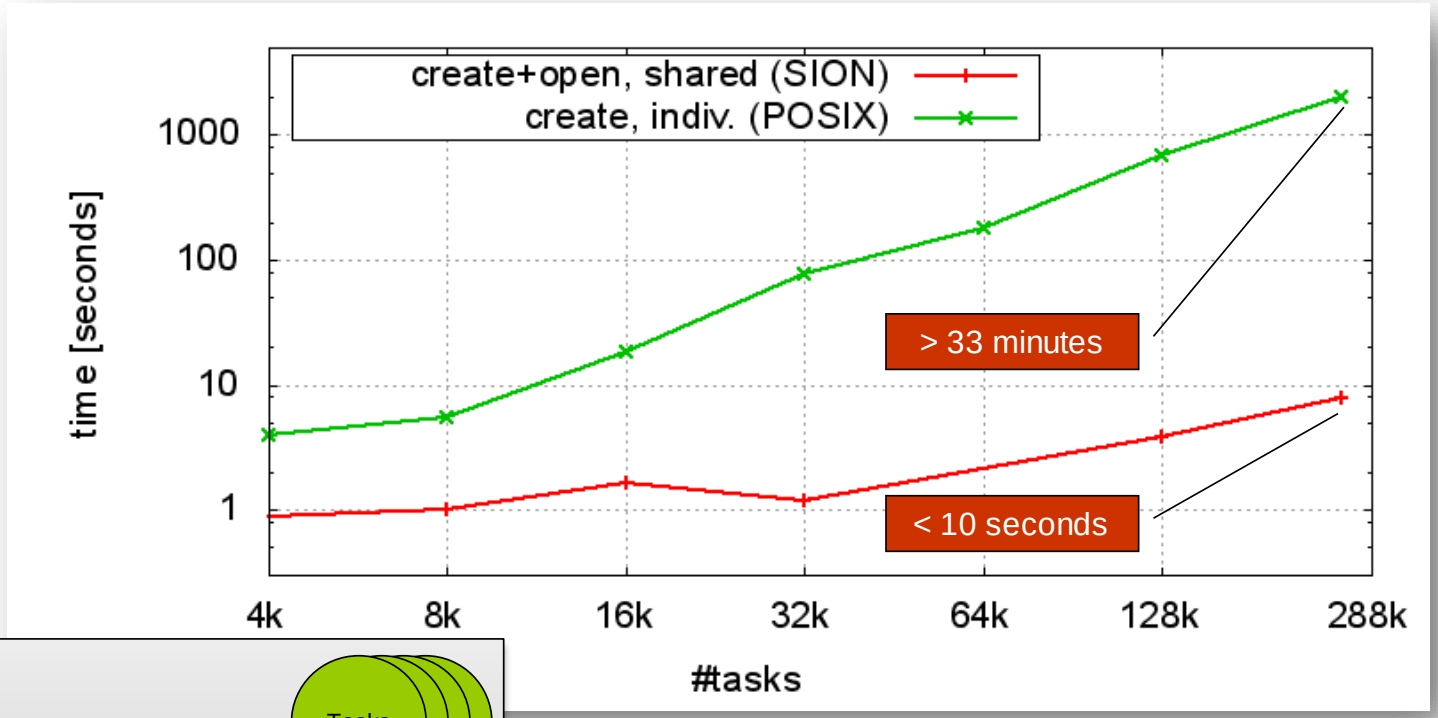
Bottlenecks:

- File creation
- File management

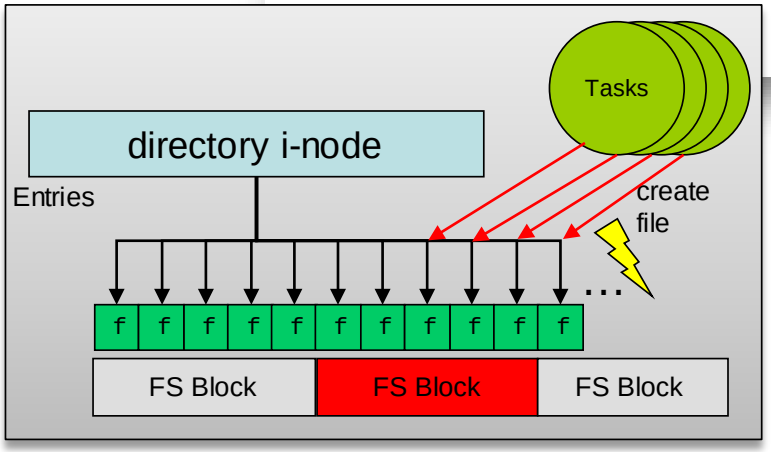
→ #files: $O(10^5)$



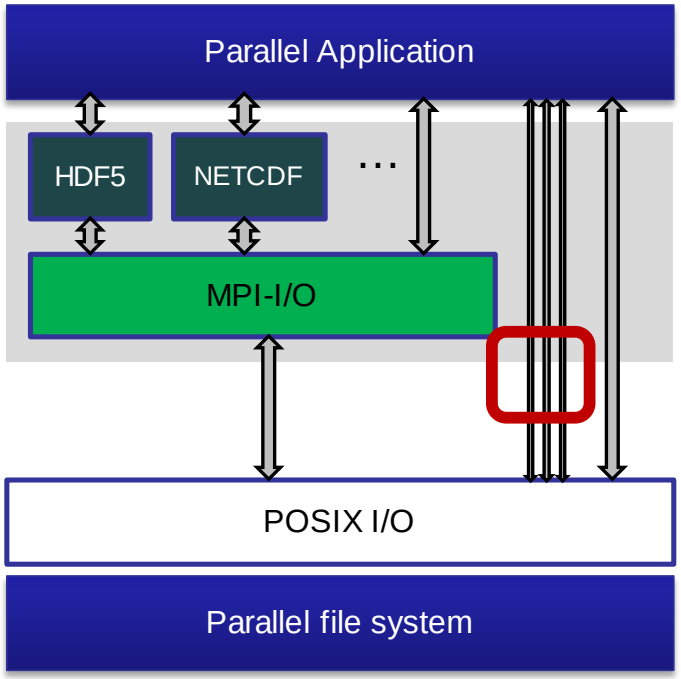
The Showstopper for Task-local I/O: Parallel Creation of Individual Files

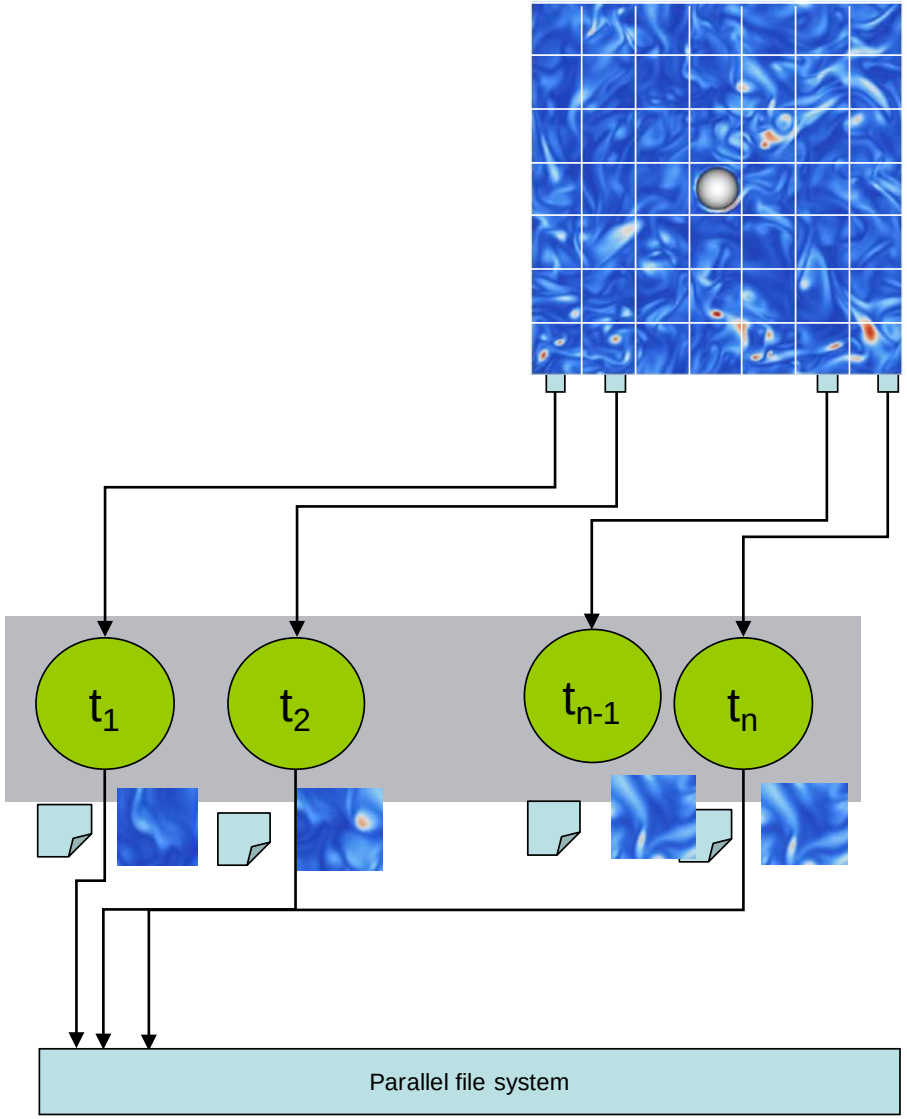


Jugene + GPFS: file create+open, one file per task versus one file per I/O-node



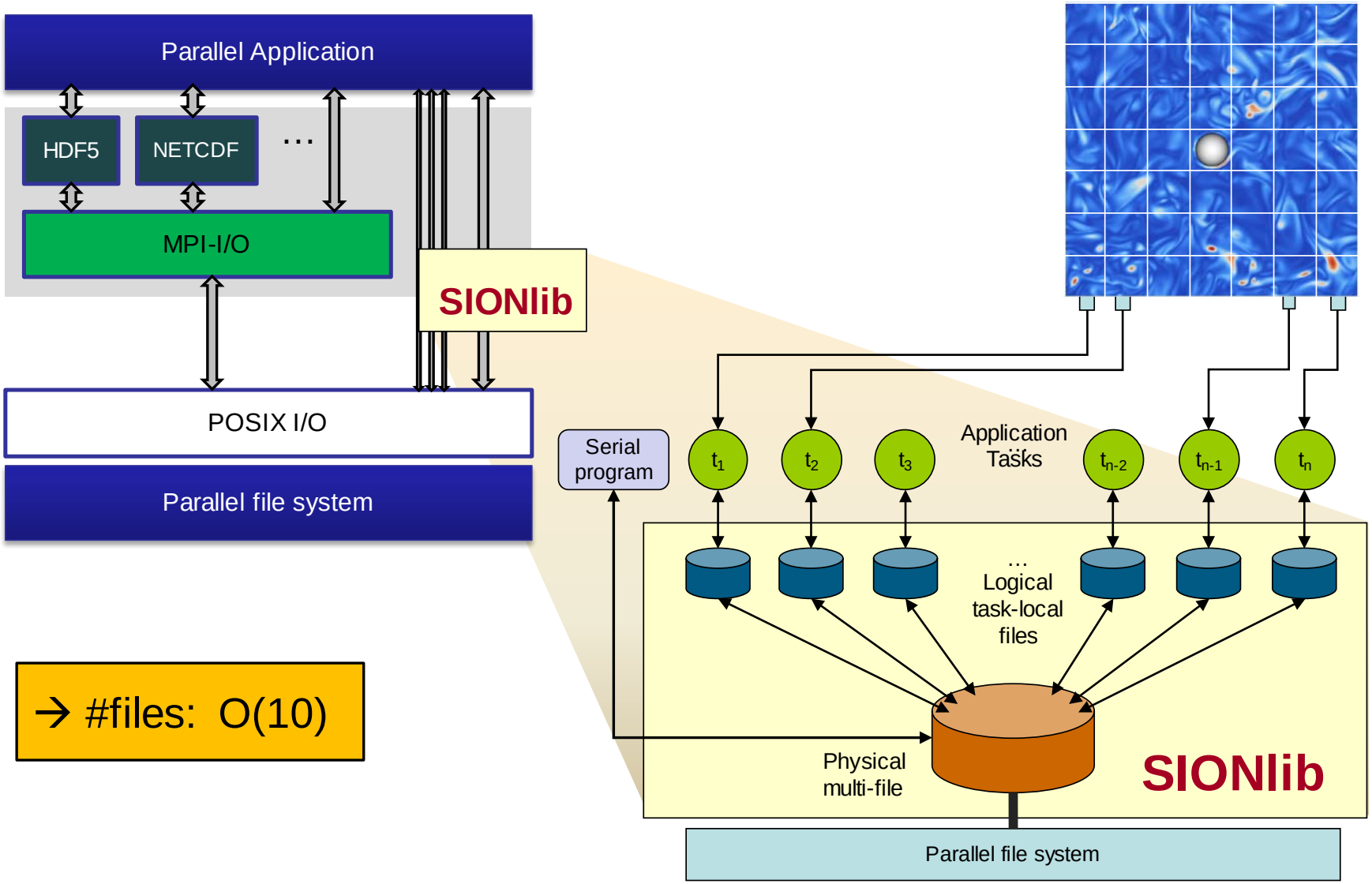
SIONlib: Shared Files for Task-local Data





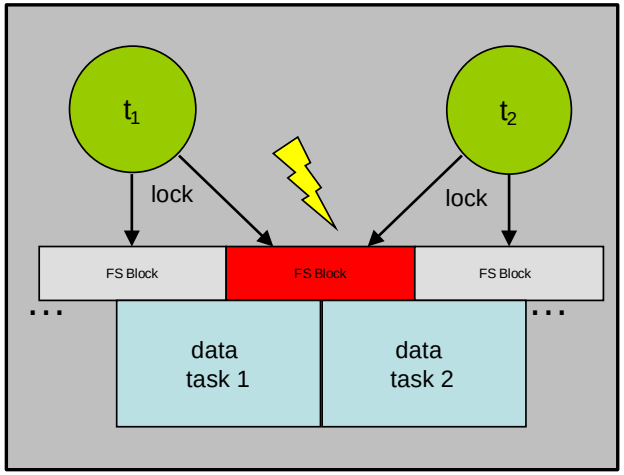
→ #files: $O(10^5)$

SIONlib: Shared Files for Task-local Data



→ #files: O(10)

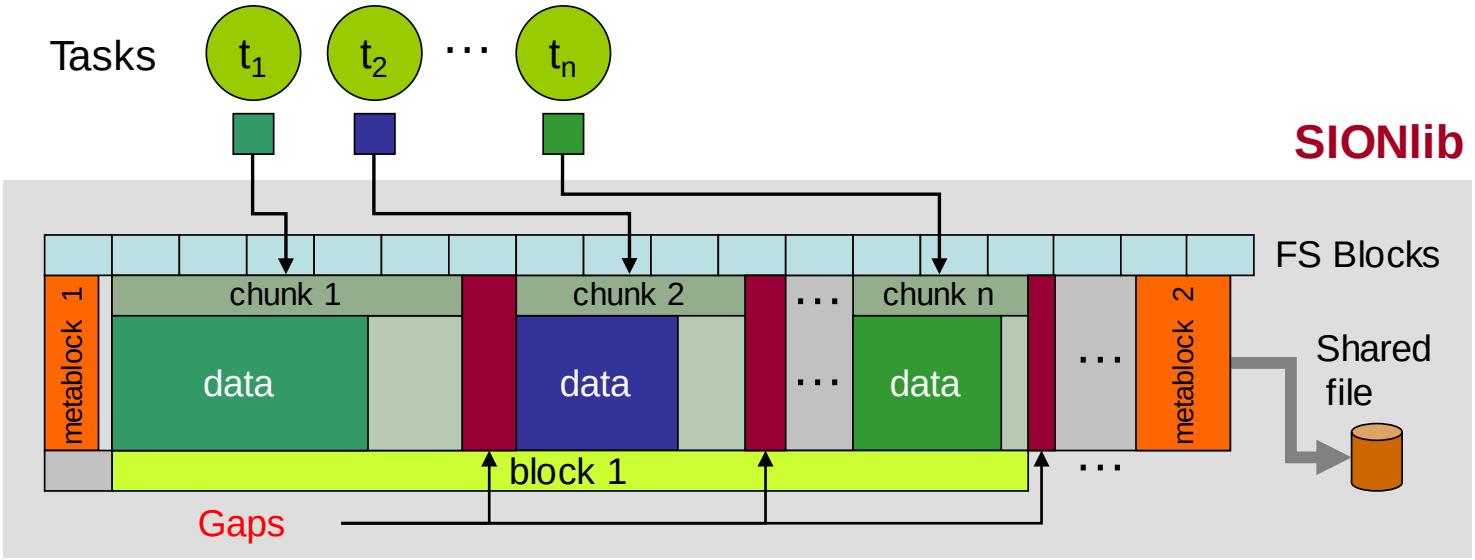
The Showstopper for Shared File I/O: … Concurrent Access & Contention



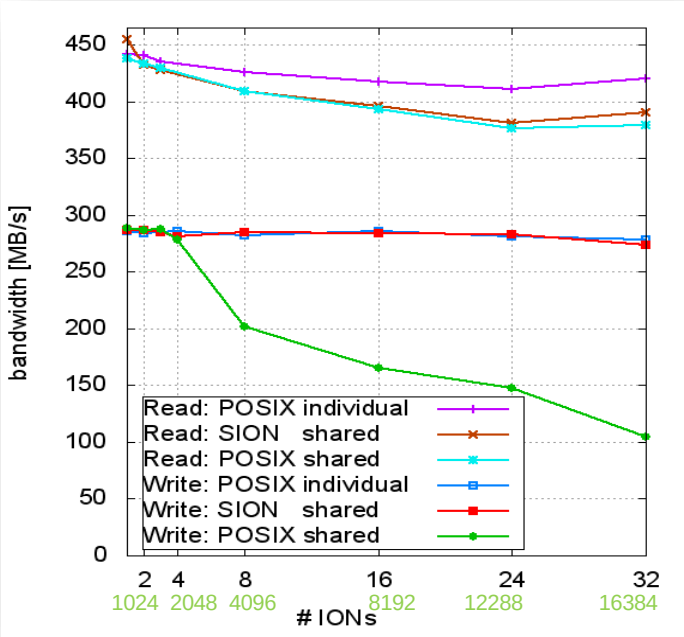
File System Block Locking → Serialization

SIONlib: Logical partitioning of Shared File:

- Dedicated data chunks per task
- Alignment to boundaries of file system blocks → no contention

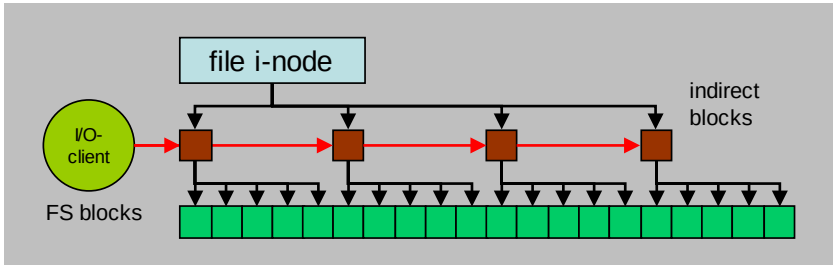


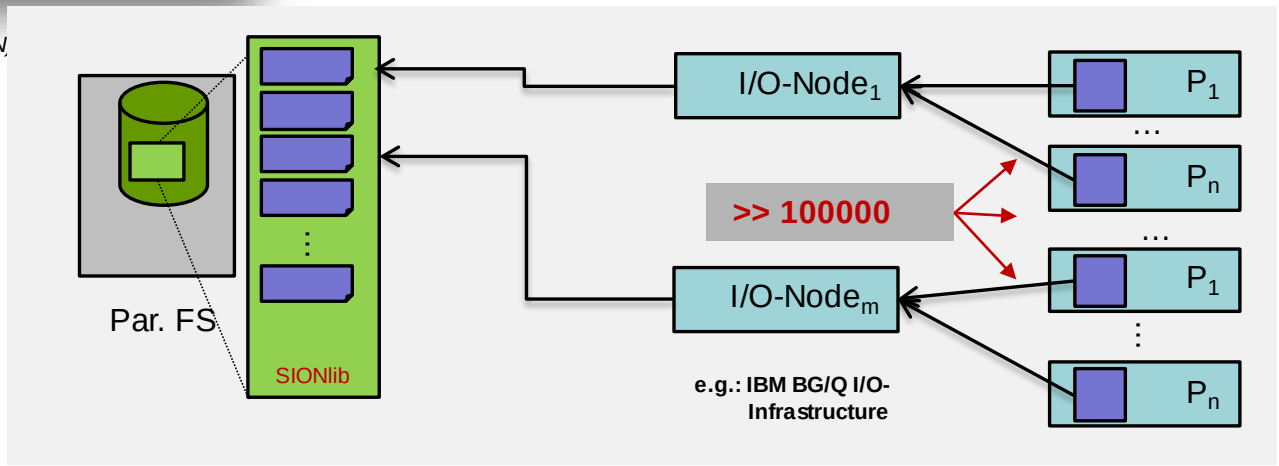
Are there more Bottlenecks?
… Increasing #tasks further …



JUGENE: Bandwidth per ION, comparison individual files (POSIX), one file per ION (SION) and one shared file (POSIX)

- Bottleneck: file meta data management
- by first GPFS client which opened the file





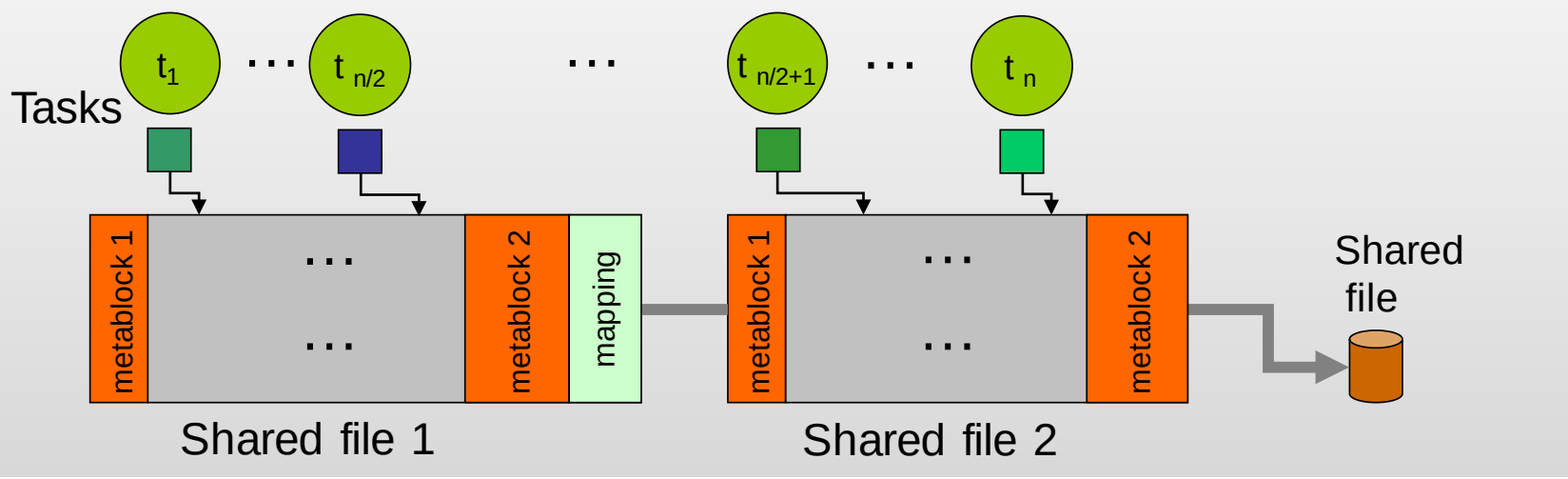
SIONlib: Multiple Underlying Physical Files

- Parallelization of file meta data handling using multiple physical files
- Mapping: Files : Tasks

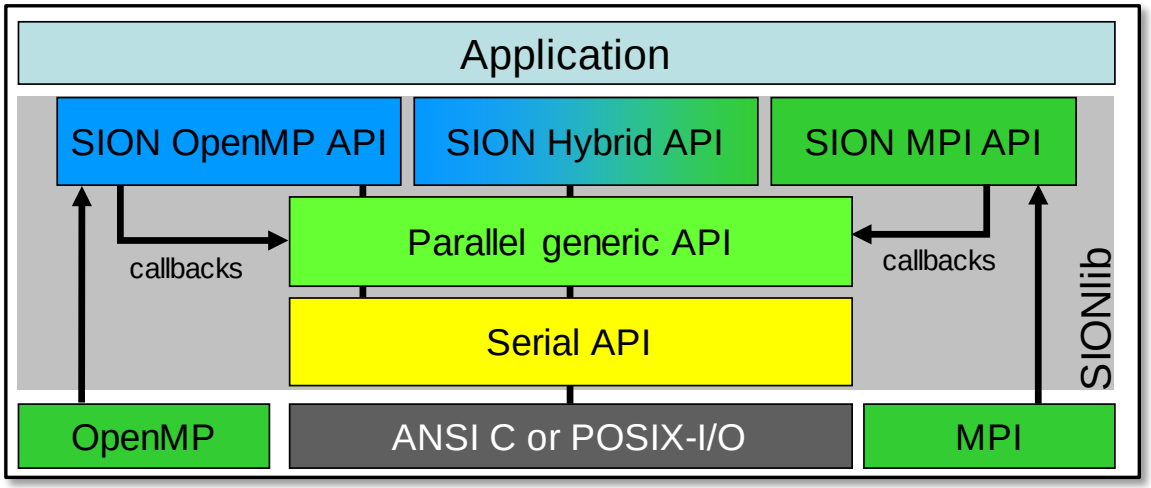
1 : n ← **p : n** → n : n

e.g. IBM Blue Gene: One file per I/O-node

e.g. Lustre: One file per OST



SIONlib: Architecture & Example



- Extension of I/O-API (ANSI C or POSIX)
- C and Fortran bindings, implementation language C
- Current versions: 1.4p3, 1.5p1
- New features: Generic API, Mapped Access, file coalescing, Key-value access
- Open source license: http://www.fz-juelich.de/jsc/sionlib

```
/* fopen() → */
sid=sion_paropen_mpi( filename , “bw“,
                      &numfiles, &chunksize,
                      gcom, &lcom, &fileptr, ...);

/* fwrite(bindata,1,nbytes, fileptr) → */

sion_fwrite(bindata,1,nbytes, sid);

/* fclose() → */
sion_parclose_mpi(sid)
```

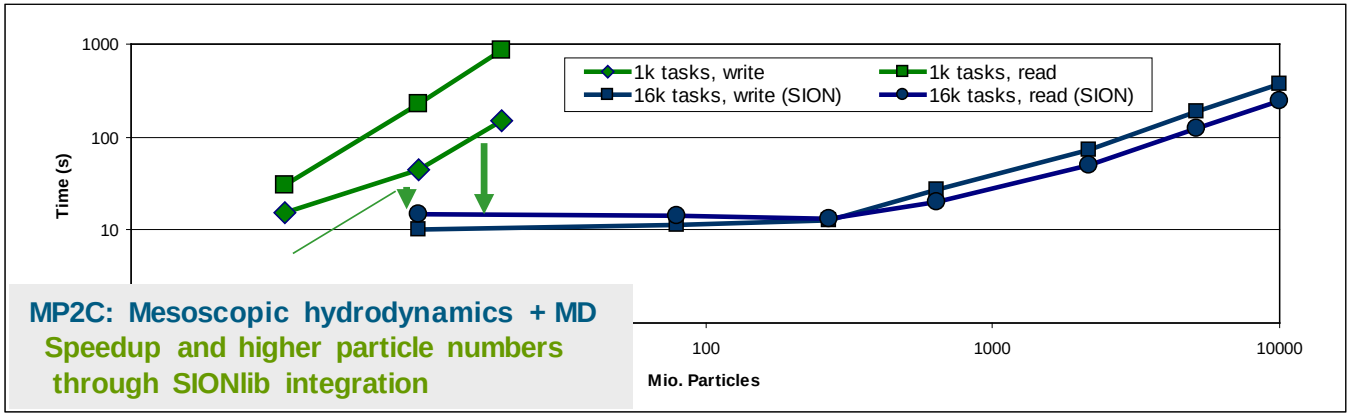
Reference: Wolfgang Frings, Felix Wolf, Ventsislav Petkov, **Scalable Massively Parallel I/O to Task-Local File**, Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Portland, Oregon, November 14-20, 2009, SC'09, New York, ACM, ISBN 978-1-60558-744-8.

SIONlib: Applications

- **Applications**

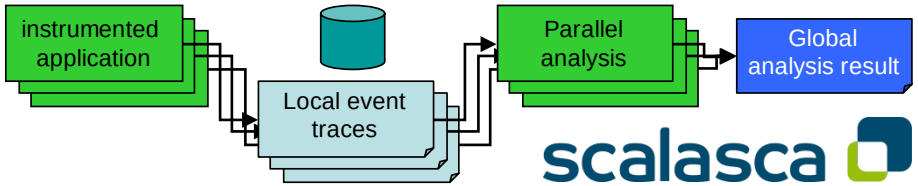
DUNE-ISTL (Multigrid solver, Univ. Heidelberg) **ITM** (Fusion-community),
LBM (Fluid flow/mass transport, Univ. Marburg), **PSC** (particle-in-cell code),
OSIRIS (Fully-explicit particle-in-cell code), **PEPC** (Pretty Efficient Parallel C. Solver)
Profasi: (Protein folding and aggr. simulator) **NEST** (Human Brain Simulation)

MP2C:



- **Tools/Projects**

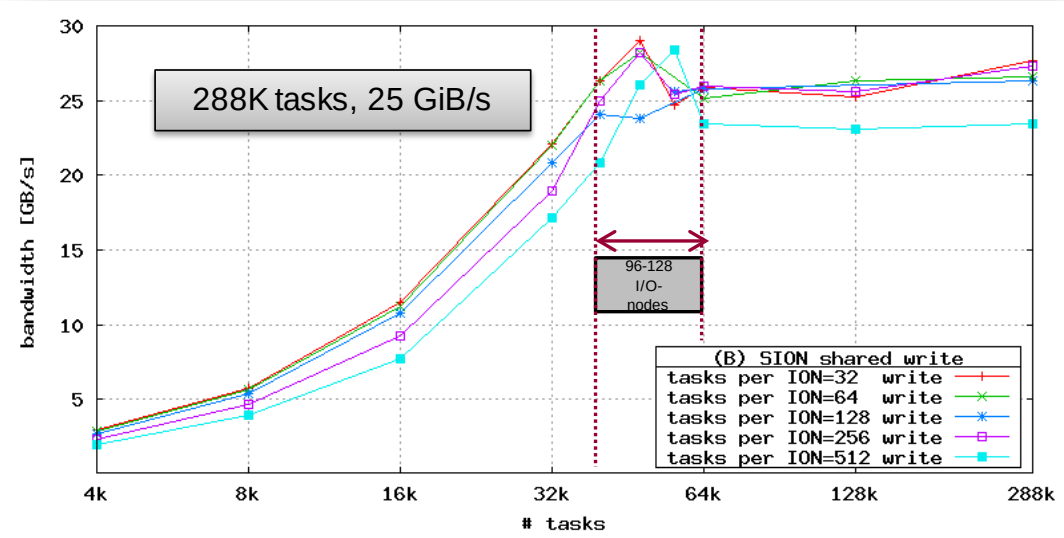
Scalasca: Performance Analysis



Score-P: Scalable Performance Measurement Infrastructure for Parallel Codes

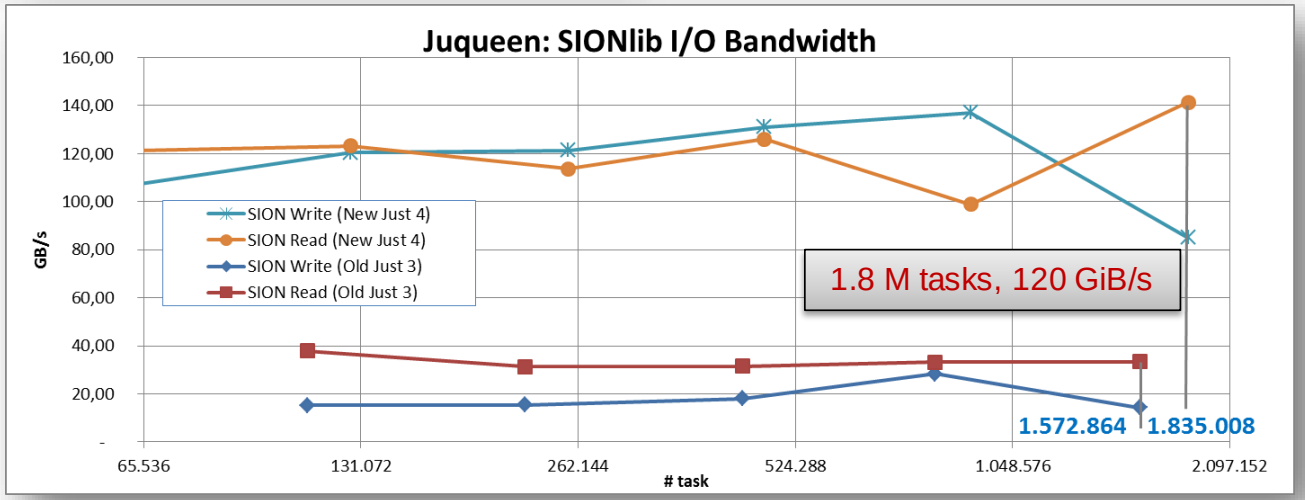
DEEP-ER: Adaption to new platform and parallelization paradigm

SIONlib: Scaling to Large # of Tasks



***JUGENE**: Total bandwidth (write), one file per I/O-node (ION), varying the number of tasks doing the I/O*

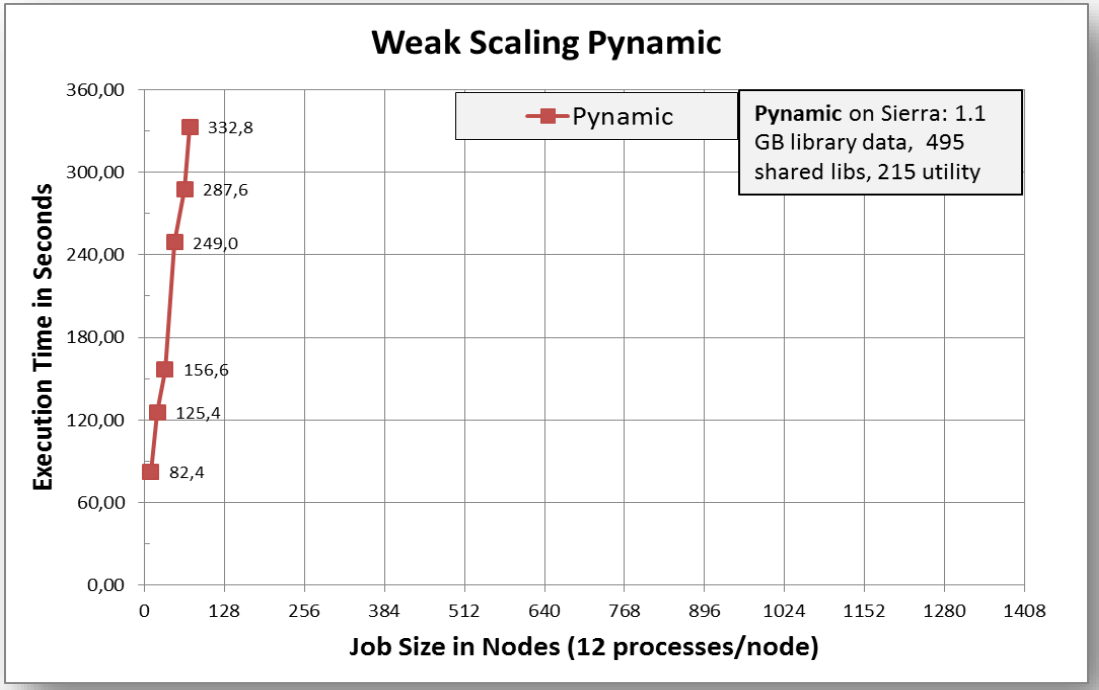
***JUQUEEN**: Total bandwidth (write/read) , one file per I/O-bridge (IOB), Old (Just3) vs. New (Just4, GSS) GPFS file system*



Spindle: Dynamic Linking and Loading Causes Major Disruption at Large Scale

- **Collaboration with LLNL**

- **Multi-physics applications at LLNL**
  - 848 shared library files
  - Load time on BG/P:
    2k tasks → 1 hour
    16k tasks → 10 hours
- **Pynamic**
  - LLNL Benchmark
  - Loads shared libraries and python files
  - 495 shared objects → 1.1 GB



Reference: Wolfgang Frings, Dong Ahn, Matthew LeGendre, Todd Gamblin, ronis de Suspinski, Felix Wolf, **Massively Parallel Loading**, In Proc. of the 27th International Conference on Supercomputing, Eugene, OR, USA, pages 389–398, ACM, 2013.

Spindle part: LLNL-PRES-638575
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Challenges Mainly Arise from File Access Storms

- Caused by dynamic linker
  - **searching** and
  - **loading** dynamic linked libraries
- Formulas:

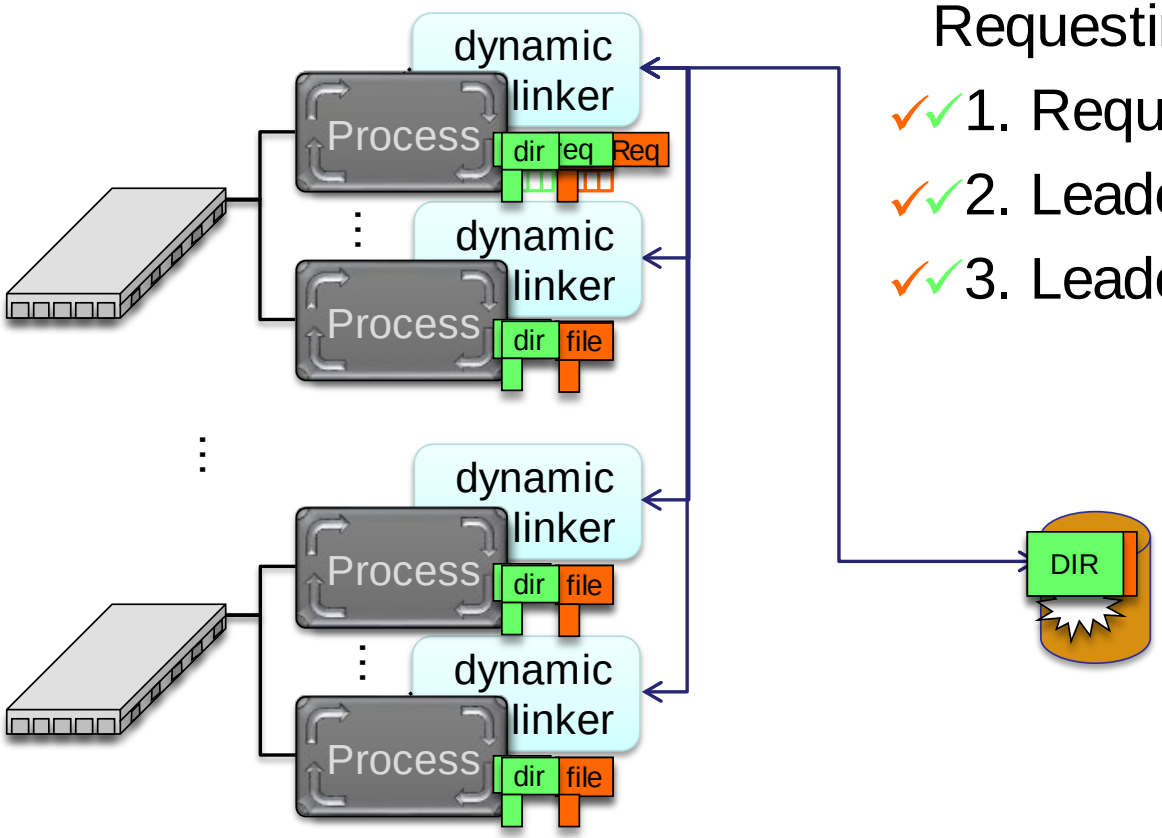
File metadata operations:
# of tests = # of processes x # of locations x # of libraries

File read operations:
# of reads = # of processes x # of libraries

- Example: Pynamic Benchmark on Sierra-Cluster
  - serial (1 task): 5,671 open/stat calls
  - parallel (23,328 tasks) : 132,293,088 open/stat calls

- Loading is nearly unchanged since 1964 (MULTICS)
- ld-linux.so uses serial POSIX file operations that are not coordinated among process.

How SPINDLE Works



Requesting dir/file:

✓✓1. Request from leader

✓✓2. Leader reads from disk

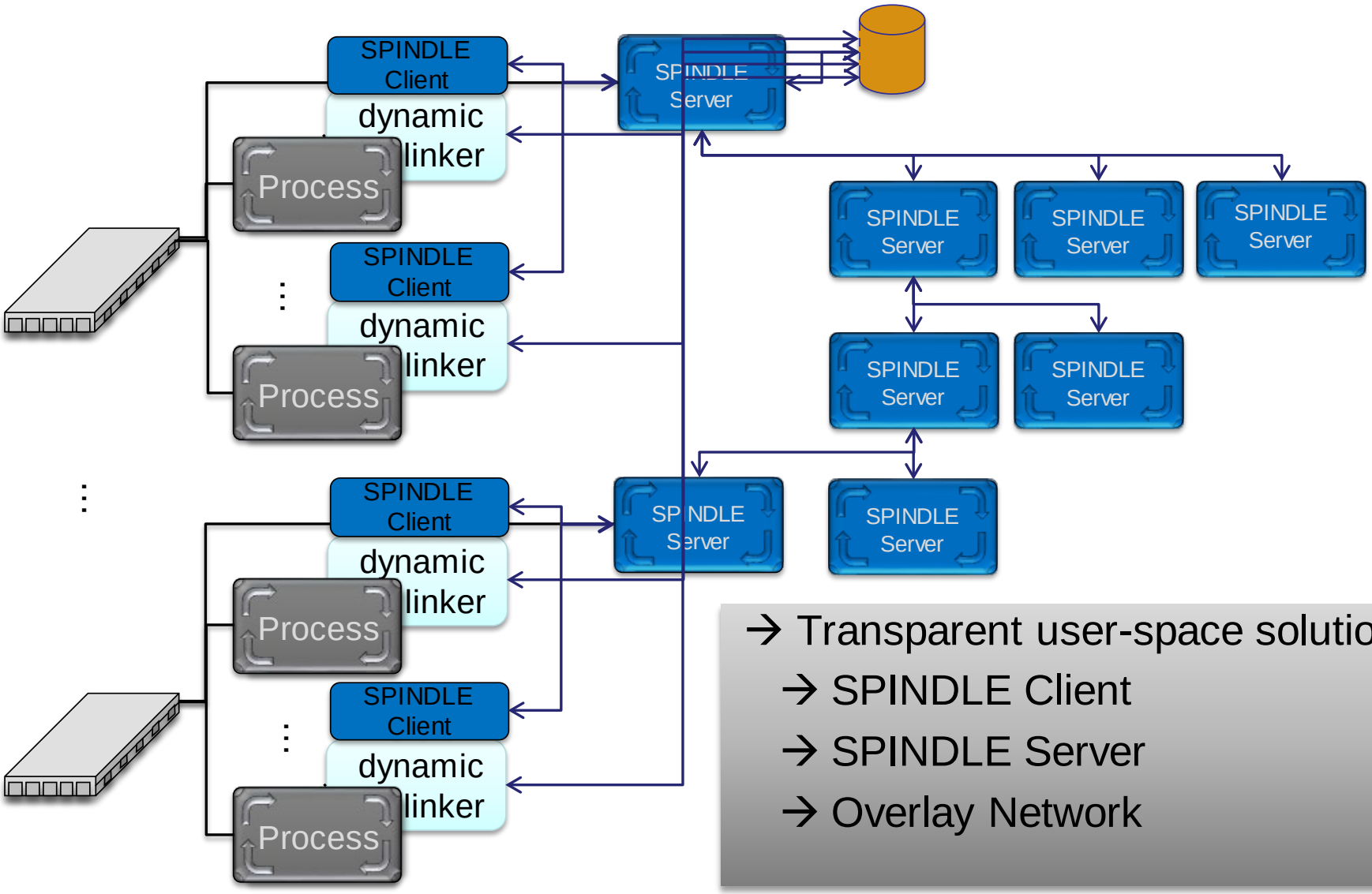
✓✓3. Leader distributes to peers

File metadata operations:
# of tests = # of locations

File read operations:
# of reads = # of libraries

SPINDLE: *Scalable Parallel Input Network for Dynamic Load Environments*

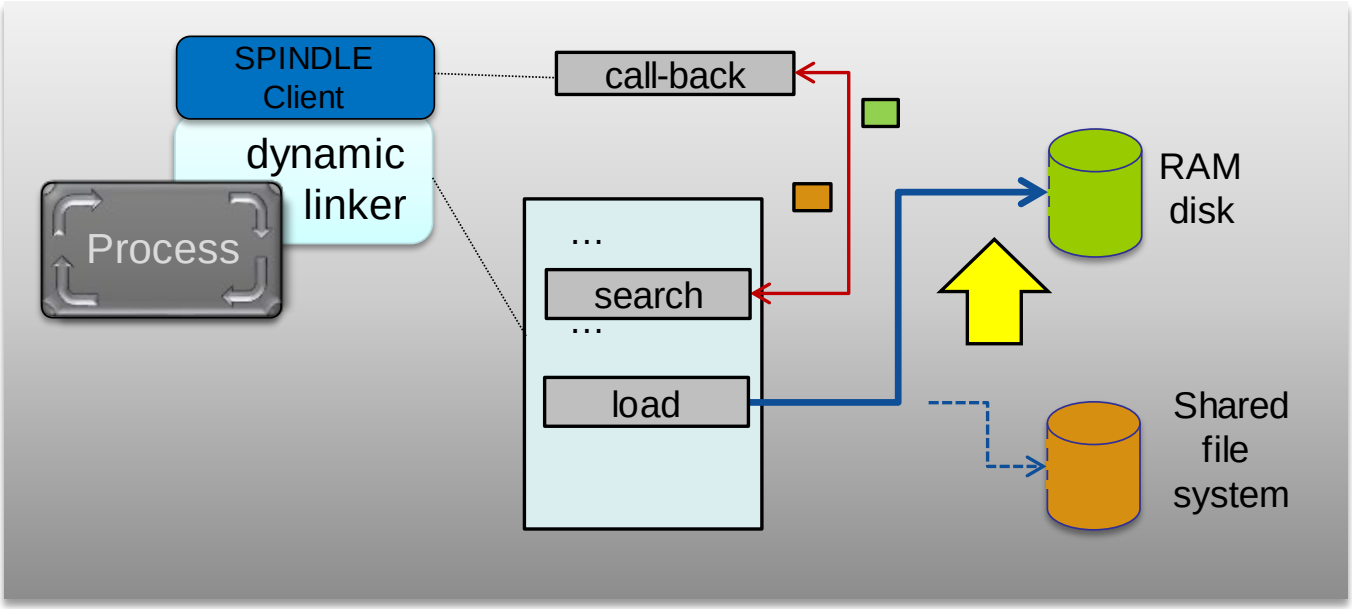
SPINDLE Components



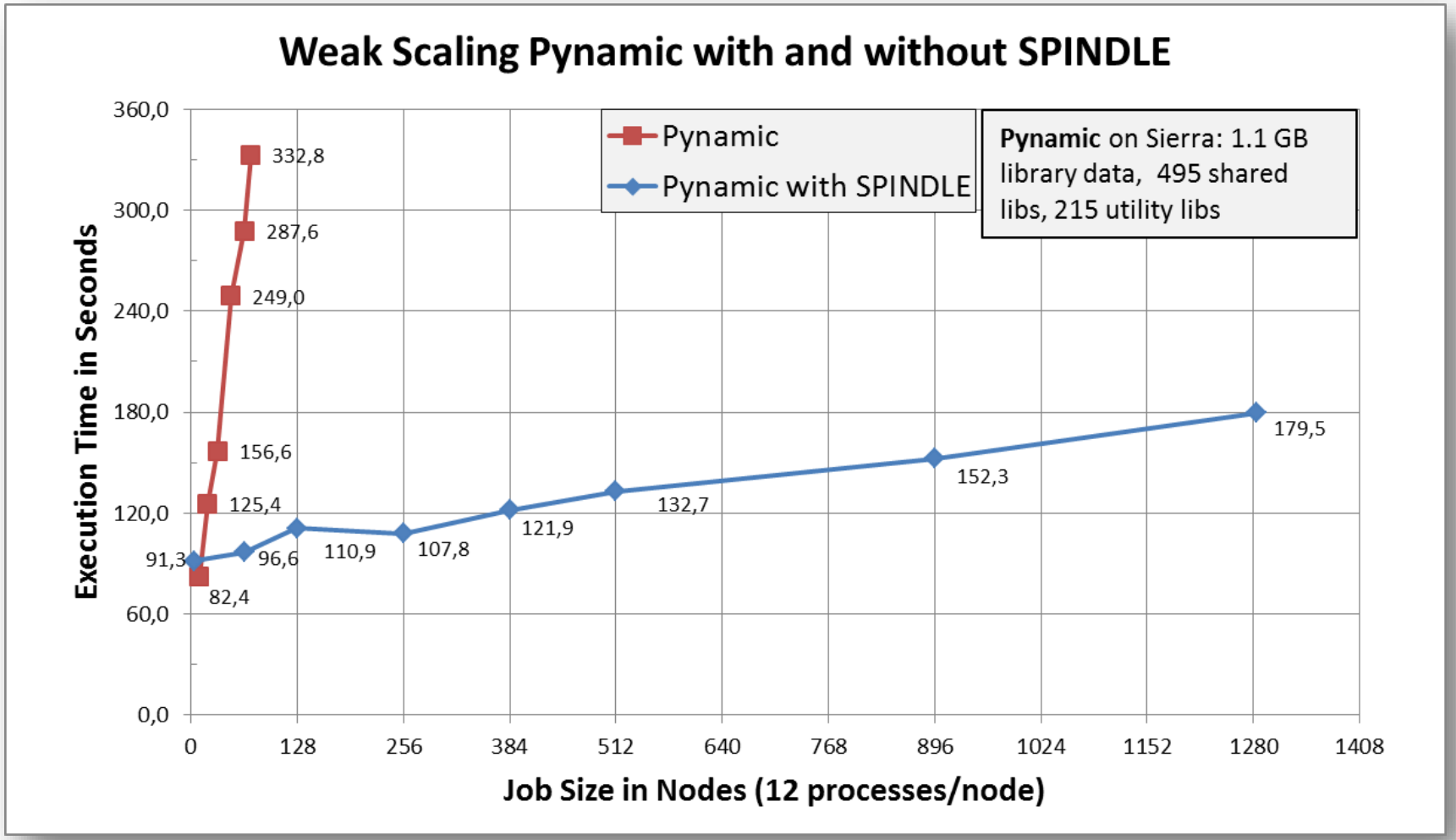
→ Transparent user-space solution
- → SPINDLE Client
- → SPINDLE Server
- → Overlay Network

SPINDLE Client intercepts Dynamic Loader transparently

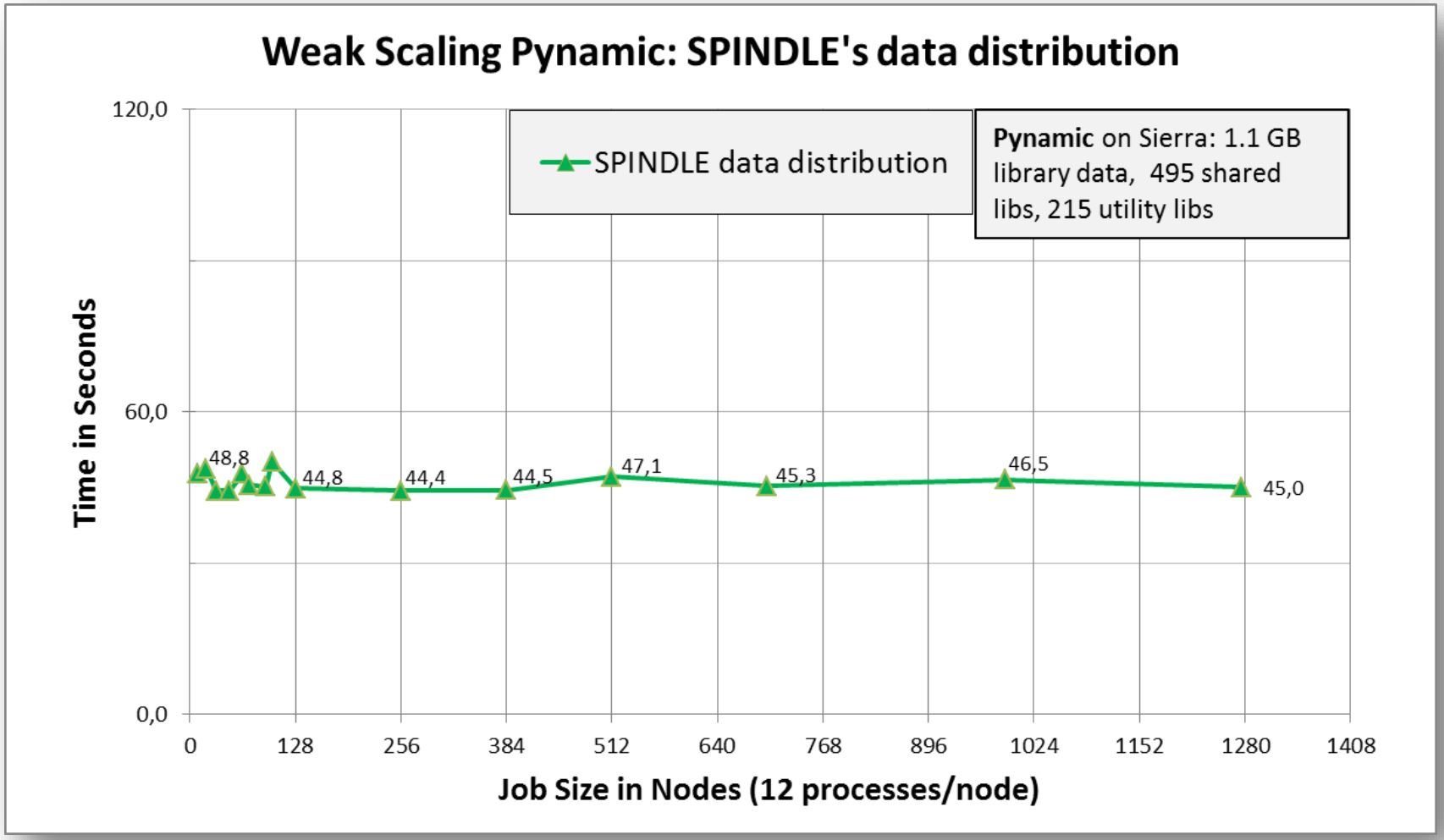
- Interception: rtld-audit interface of GNU linker
  - User-space definition of call-back functions
  - Redirect remote file system loads to a local RAM disk



SPINDLE's Performance



Constant Overhead of SPINDLE's Data Distribution



Launching SPINDLE

- SPINDLE wrapper call:

```
% spindle srun -n 512 myapp.exe <args>
```

- Executable is not modified
- SPINDLE scalably loads:
  - Library files (from dependencies and dlopen)
  - Executable
  - Python .py/.pyc/.pyo files
  - exec/execv/execve/… call targets
- Can follow forked processes
- Integrated with LaunchMON (http://sourceforge.net/projects/launchmon)

Availability & Outlook

- Availability of SPINDLE
  - WWW: https://computation-rnd.llnl.gov/spindle
  - GitHub: https://github.com/hpc/Spindle
    Licence: GNU Lesser General Public License (LGPL)
  - Build: Configure and Libtool, Test Environment
  - Platforms: Linux/x86_64 cluster using SLURM
  - Version: 0.9
- SPINDLE's next steps
  - Porting, optimization and customization for broader range of HPC systems: IBM Blue Gene using I/O-nodes
  - Tighter integration into various HPC system software (resource manager software systems)

Conclusion

- HPC I/O at large scale:
  - Meta-data bottlenecks caused by the large number program instances issuing I/O-operations

- Two Solutions:

  SIONlib: Parallel I/O to task-local files at large scale

  Spindle: Scalable Parallel Input Network for Dynamic Load Environments

- Methods:
  - Reducing the number of file system objects (e.g. task-local files) by using file container (SIONlib)
  - Reducing the number of file system accesses by using a cache server overlay network (Spindle)